Docket No. 15437-0598 *Patent*

UNITED STATES PATENT APPLICATION

FOR

SYSTEM STATISTICS VIRTUALIZATION FOR OPERATING SYSTEM PARTITIONS

INVENTORS:

OZGUR C. LEONARD
ANDREW G. TUCKER

PREPARED BY:

HICKMAN PALERMO TRUONG & BECKER LLP
1600 WILLOW STREET
SAN JOSE, CALIFORNIA 95125
(408) 414-1080

"Express Mail" mailing label number EV323351595US

Date of Deposit January 20, 2004

SYSTEM STATISTICS VIRTUALIZATION FOR OPERATING SYSTEM PARTITIONS

Inventors: OZGUR C. LEONARD, ANDREW G. TUCKER

Claim Of Priority

[0001] This application claims priority to U.S. Provisional Application Serial No. 60/469,558, filed May 9, 2003, entitled "OPERATING SYSTEM VIRTUALIZATION," by Andrew G. Tueker, et al., the entire contents of which are incorporated by reference as if fully set forth herein.

Background

[0002] Many of today's computing systems include computing resources that are not fully utilized. The owners of these systems often could benefit by increasing the utilization of these systems' computing resources.

[0003] A number of approaches could be adopted in order to increase utilization. Under a "consolidation" approach, the processes and data of multiple parties might be co-located on a single hardware unit in order to more fully utilize the resources of the hardware unit. Under the consolidation approach, multiple parties might share a single hardware unit's resources, including file systems, network connections, and memory structures. For example, multiple businesses might have separate websites that are hosted by the same server.

[0004] However, some of the parties might not know or trust each other. In some cases, some of the parties actually might be competitors with others of the parties. Under such circumstances, each party would want to ensure that its processes and data were shielded, or isolated, from access by other parties and those other parties' processes.

[0005] Mechanisms that would isolate one party's processes and data from other parties sharing the same hardware unit have been proposed. For example, a "jail" mechanism provides the ability to partition an operating system environment ("OSE") into a "non-jailed" environment and one or more "jailed" environments. The jail mechanism allows users, processes, and data to be associated with a jailed environment. For example, one group of users, processes, and data may be associated with one jailed environment, and another group of users, processes, and data may be associated with another jailed environment. The jail mechanism restricts users and processes that are associated with a particular jailed environment from accessing processes and data that are associated with environments (both jailed and non-jailed) other than the particular jailed environment.

[0006] Some OSEs provide a statistics recording mechanism that records statistics that pertain to one or more resources that are defined within the OSE. For example, the resources defined within an OSE might include one or more separate Network File System ("NFS") mounts. Each NFS mount might be associated with a separate statistical data structure. When data is written to a particular NFS mount, the statistics recording mechanism might update, for example, information in the particular NFS mount's associated statistical data structure. The information might include, for example, an indication of the quantity of data that has been written to the particular NFS mount.

[0007] As discussed above, an OSE may be partitioned into a non-jailed environment and one or more jailed environments. However, no previous statistics recording approaches contemplated multiple partitions within an OSE. Because previous statistics recording approaches did not contemplate a partitioned OSE, previous statistics recording approaches do not account for certain security and scope issues that do not arise in non-partitioned OSEs.

Summary

[0008] In accordance with one embodiment of the present invention, a mechanism is disclosed for virtualizing system statistics in an OSE that has been partitioned into a global zone and one or more non-global zones. The OSE comprises one or more processes and one or more system resources. Each process is associated with a zone in which that process executes. Each system resource may be associated with a statistical data structure. Statistical data about a resource is stored in that resource's associated statistical data structure. Each statistical data structure is associated with at least one zone and one or more key values.

[0009] According to one aspect, a process may specify one or more key values. The kernel responsively selects, from among a plurality of statistical data structures, a set of statistical data structures that are associated with the specified key values. The kernel determines whether any statistical data structure in the set is associated with the zone in which the process executes. If a statistical data structure in the set is associated with the zone in which the process executes, then the kernel sends, to the process, statistical data that is stored in that statistical data structure.

[0010] According to one aspect, a process may request a list of statistical data structures. The kernel responsively selects, from among a plurality of statistical data structures, a set of statistical data structures that are associated with the zone in which the process executes. The kernel sends, to the process, a list of the statistical data structures in the set. The set does not contain any statistical data structures that are not associated with the zone in which the process executes.

Brief Description of the Drawings

[0011] Fig. 1 is a functional block diagram of a representative OSE for a computing system in which one embodiment of the present invention may be implemented.

[0012] Fig. 2A depicts an overview of an operational flow for virtualizing system statistics in an OSE that has been partitioned into a global zone and one or more non-global zones, in accordance with one embodiment of the present invention.

[0013] Fig. 2B depicts an overview of an operational flow for constructing a virtualized list of statistical data structures in an OSE that has been partitioned into a global zone and one or more non-global zones, in accordance with one embodiment of the present invention.

[0014] Fig. 3A depicts an operational flow for virtualizing system statistics in an OSE that has been partitioned into a global zone and one or more non-global zones, in accordance with one embodiment of the present invention.

[0015] Fig. 3B depicts an operational flow for constructing a virtualized list of statistical data structures in an OSE that has been partitioned into a global zone and one or more non-global zones, in accordance with one embodiment of the present invention.

[0016] Fig. 4 is a hardware block diagram of a sample computer system, upon which one or more components of an embodiment of the present invention may be implemented.

Detailed Description of Embodiment(s)

Overview

[0017] Fig. 1 illustrates a functional block diagram of an OSE 100 in accordance with one embodiment of the present invention. OSE 100 may be derived by executing an operating system (OS) in a general-purpose computer system, such as computer system 400 illustrated in Fig. 4, for example. Although Fig. 4 depicts a system that contains centralized component resources, embodiments may be implemented on systems that comprise remotely distributed component resources (e.g., processors, memory, persistent storage, etc.) that access each other via a network. For illustrative purposes, the OS is assumed to be Solaris™ manufactured by Sun Microsystems, Inc. of Santa Clara, California. However, the concepts taught herein may be applied to any OS, including but not limited to Unix, Linux, Microsoft Windows, MacOS, etc.

[0018] As shown in Fig. 1, OSE 100 may comprise one or more zones (also referred to herein as partitions), including a global zone 130 and zero or more non-global zones 140. The global zone 130 is the general OSE that is created when the OS is booted and executed, and serves as the default zone in which processes may be executed if no non-global zones 140 are created. In the global zone 130, administrators and/or processes having the proper rights and privileges can perform generally any task and access any device/resource that is available on the computer system on which the OS is run. Thus, in the global zone 130, an administrator can administer the entire computer system. In one embodiment, it is in the global zone 130 that an administrator executes processes to configure and to manage the non-global zones 140.

[0019] The non-global zones 140 represent separate and distinct partitions of the OSE 100. Each of non-global zones 140 may be viewed as a virtual operating system environment

("VOSE"). One of the purposes of the non-global zones 140 is to provide isolation. In one embodiment, a non-global zone 140 can be used to isolate a number of entities, including but not limited to processes 170, one or more file systems 180, and one or more logical network interfaces 182. Because of this isolation, processes 170 executing in one non-global zone 140 cannot access or affect processes in any other zone. Similarly, processes 170 in a non-global zone 140 cannot access or affect the file system 180 of another zone, nor can they access or affect the network interface 182 of another zone. As a result, the processes 170 in a non-global zone 140 are limited to accessing and affecting the processes and entities in that zone. Isolated in this manner, each non-global zone 140 behaves like a virtual standalone computer. While processes 170 in different non-global zones 140 cannot access or affect each other, it should be noted that they may be able to communicate with each other via a network connection through their respective logical network interfaces 182. This is similar to how processes on separate standalone computers communicate with each other.

[0020] Having non-global zones 140 that are isolated from each other may be desirable in many applications. For example, if a single computer system running a single instance of an OS is to be used to host applications for different competitors (e.g., competing websites), then it would be desirable to isolate the data and processes of one competitor from the data and processes of another competitor. That way, it can be ensured that information will not be leaked between the competitors. Partitioning an OSE 100 into non-global zones 140 is one possible way of achieving this isolation. Competing applications (e.g., websites) may then be hosted in separate non-global zones 140.

[0021] In one embodiment, each non-global zone 140 may be administered separately. More specifically, it is possible to assign a zone administrator to a particular non-global zone 140 and grant that zone administrator rights and privileges to manage various aspects of that

non-global zone 140. With such rights and privileges, the zone administrator can perform any number of administrative tasks that affect the processes and other entities within that non-global zone 140. However, the zone administrator cannot change or affect anything in any other non-global zone 140 or the global zone 130. Thus, in the above example, each competitor can administer his/her zone, and hence, his/her own set of applications, but cannot change or affect the applications of a competitor. In one embodiment, to prevent a non-global zone 140 from affecting other zones, the entities in a non-global zone 140 generally are not allowed to access or control any of the physical devices of the computer system.

[0022] In contrast to a non-global zone administrator, a global zone administrator with proper rights and privileges may administer all aspects of the OSE 100 and the computer system as a whole. Thus, a global zone administrator may, for example, access and control physical devices, allocate and control system resources, establish operational parameters, etc. A global zone administrator may also access and control processes and entities within a non-global zone 140.

[0023] In one embodiment, kernel 150 enforces the zone boundaries. More specifically, kernel 150 ensures that processes 170 in one non-global zone 140 are not able to access or affect processes 170, file systems 180, and network interfaces 182 of another zone (non-global or global). In addition to enforcing the zone boundaries, kernel 150 also provides a number of other services. These services include but are not limited to mapping the network interfaces 182 of the non-global zones 140 to the physical network devices 120 of the computer system, and mapping the file systems 180 of the non-global zones 140 to an overall file system and a physical storage 110 of the computer system.

Non-Global Zone States

[0024] In one embodiment, a non-global zone 140 may take on one of four states: (1) Configured; (2) Installed; (3) Ready; and (4) Running. When a non-global zone 140 is in the Configured state, it means that an administrator in the global zone 130 has invoked an operating system utility (in one embodiment, zonecfg(1m)) to specify all of the configuration parameters of a non-global zone 140, and has saved that configuration in persistent physical storage 110. In configuring a non-global zone 140, an administrator may specify a number of different parameters. These parameters may include, but are not limited to, a zone name, a zone path to the root directory of the zone's file system 180, specification of one or more file systems to be mounted when the zone is created, specification of zero or more network interfaces, specification of devices to be configured when the zone is created, and zero or more resource pool associations.

[0025] Once a zone is in the Configured state, a global administrator may invoke another operating system utility (in one embodiment, zoneadm(1m)) to put the zone into the Installed state. When invoked, the operating system utility interacts with the kernel 150 to install all of the necessary files and directories into the zone's root directory, or a subdirectory thereof.

[0026] To put an Installed zone into the Ready state, a global administrator invokes an operating system utility (in one embodiment, zoneadm(1m) again), which a zoneadmd process 162 causes to be started (there is a zoneadmd process associated with each non-global zone). In one embodiment, zoneadmd 162 runs within the global zone 130 and is responsible for managing its associated non-global zone 140. After zoneadmd 162 is started, it interacts with the kernel 150 to establish the non-global zone 140. In creating a non-global zone 140, a number of operations are performed, including but not limited to assigning a zone ID, starting a zsched process 164 (zsched is a kernel process; however, it runs within

the non-global zone 140, and is used to track kernel resources associated with the non-global zone 140), mounting file systems 180, plumbing network interfaces 182, configuring devices, and setting resource controls. These and other operations put the non-global zone 140 into the Ready state to prepare it for normal operation.

[0027] Putting a non-global zone 140 into the Ready state gives rise to a virtual platform on which one or more processes may be executed. This virtual platform provides the infrastructure necessary for enabling one or more processes to be executed within the non-global zone 140 in isolation from processes in other non-global zones 140. The virtual platform also makes it possible to isolate other entities such as file system 180 and network interfaces 182 within the non-global zone 140, so that the zone behaves like a virtual standalone computer. When a non-global zone 140 is in the Ready state, no user or non-kernel processes are executing inside the zone (as is mentioned above, zsched is a kernel process, not a user process). Thus, the virtual platform provided by the non-global zone 140 is independent of any processes executing within the zone. Put another way, the zone and hence, the virtual platform, exists even if no user or non-kernel processes are executing within the zone. This means that a non-global zone 140 can remain in existence from the time it is created until either the zone or the OS is terminated. The life of a non-global zone 140 need not be limited to the duration of any user or non-kernel process executing within the zone.

[0028] After a non-global zone 140 is in the Ready state, it can be transitioned into the Running state by executing one or more user processes in the zone. In one embodiment, this is done by having zoneadmd 162 start an init process 172 in its associated zone. Once started, the init process 172 looks in the file system 180 of the non-global zone 140 to determine what applications to run. The init process 172 then executes those applications to

give rise to one or more other processes 174. In this manner, an application environment is initiated on the virtual platform of the non-global zone 140. In this application environment, all processes 170 are confined to the non-global zone 140; thus, they cannot access or affect processes, file systems, or network interfaces in other zones. The application environment exists so long as one or more user processes are executing within the non-global zone 140.

[0029] After a non-global zone 140 is in the Running state, its associated zoneadmd 162 can be used to manage it. Zoneadmd 162 can be used to initiate and control a number of zone administrative tasks. These tasks may include, for example, halting and rebooting the non-global zone 140. When a non-global zone 140 is halted, it is brought from the Running state down to the Installed state. In effect, both the application environment and the virtual platform are terminated. When a non-global zone 140 is rebooted, it is brought from the Running state down to the Installed state, and then transitioned from the Installed state through the Ready state to the Running state. In effect, both the application environment and the virtual platform are terminated and restarted. These and many other tasks may be initiated and controlled by zoneadmd 162 to manage a non-global zone 140 on an ongoing basis during regular operation.

Overview of Virtualized System Statistics

[0030] Fig. 2A depicts an overview of an operational flow for virtualizing system statistics in an OSE that has been partitioned into a global zone and one or more non-global zones, in accordance with one embodiment of the present invention. In block 202, one or more key values are received from a process that executes in one of a plurality of VOSEs (i.e., non-global zones 140). In block 204, a set of one or more statistical data structures that are associated with the one or more key values are selected from among a plurality of

statistical data structures. In block 206, it is determined whether any statistical data structure in the set of one or more statistical data structures is associated with the VOSE in which the process executes. If so, then control passes to block 208. If not, then control passes to block 210.

[0031] In block 208, statistical data that is stored in a statistical data structure that is both in the set and associated with the VOSE in which the process executes is sent to the process. Alternatively, in block 210, no statistical data that is stored in statistical data structures in the set is sent to the process. Thus, even if multiple statistical data structures (each associated with a different non-global zone) are associated with the same key values, a process may, by specifying the key values, obtain statistical data from the one of the multiple statistical data structures that is associated with the process' associated non-global zone (if that statistical data structure exists). The process does not need to specify, or even be aware of, the non-global zone in which the process executes. Furthermore, no user or mechanism needs to ensure that key values associated with a particular non-global zone's associated statistical data structures are not also associated with other non-global zones' associated statistical data structures.

[0032] Fig. 2B depicts an overview of an operational flow for constructing a virtualized list of statistical data structures in an OSE that has been partitioned into a global zone and one or more non-global zones, in accordance with one embodiment of the present invention. In block 252, a request for a list of statistical data structures is received from a process that executes in one of a plurality of VOSEs (i.e., non-global zones 140). In block 254, it is determined in which VOSE of the plurality of VOSEs the process executes. In block 256, a set of one or more statistical data structures that are associated with the VOSE in which the process executes are selected from among the plurality of statistical data structures. In block

258, a list of statistical data structures in the set of one or more statistical data structures is sent to the process. Thus, processes executing in a particular non-global zone may, without specifying the non-global zone in which those processes execute, obtain a list that contains only statistical data structures that are associated with the particular non-global zone. This enables processes to interact with the partitioned OSE's system statistics reporting mechanisms in the same manner that the processes would interact with a non-partitioned OSE's system statistics reporting mechanisms.

Process/Zone Associations

[0033] In one embodiment, when an application program is executed within a zone, an application process is started as a child process of a process executing within that zone. Because each child process is associated with the zone with which that child process' parent process is associated, the application process is associated with the zone in which it was executed. For example, if a zone administrator for non-global zone 140a executes an application program, then, in non-global zone 140a, an application process is started as a child process of a process executing in non-global zone 140a. Because the parent process of the utility process is associated with non-global zone 140a, the application process also is associated with non-global zone 140a. Similarly, if a zone administrator for global zone 130 executes an application program then, in global zone 130, an application process will be started and associated with global zone 130.

[0034] Each zone is associated with a different zone identifier. Associating a process with a zone may be accomplished, for example, by storing an association between the process' unique process identifier and the zone's unique zone identifier.

Virtualized System Statistics

[0035] A partitioned OSE should allow users and processes that are associated with a particular non-global zone to access system statistics that pertain specifically to the use of system resources by users and processes associated with the particular non-global zone. However, except for users and processes that are associated with the global zone, users and processes that are not associated with the particular non-global zone should be prevented from accessing system statistics that pertain only to the use of system resources by users and processes that are associated with the particular non-global zone. Otherwise, system security could be compromised.

[0036] Additionally, in a partitioned OSE, users and processes that are associated with a particular non-global zone typically have little use for system statistics that do not specifically pertain to the use of system resources by users and processes associated with the particular non-global zone. For example, an administrator of non-global zone 140a might find system statistics that detail the use of system resources by users and processes associated with non-global zone 140a to be much more useful than system statistics that consist solely of a non-zone-specific summary of the use of system resources generally.

[0037] Therefore, in one embodiment, a mechanism for maintaining separate zone-specific system statistics for each zone is provided. In one embodiment, kernel 150 maintains a separate statistical data structure, referred to herein as a "k-stat," for each system resource that is defined within OSE 100. For example, system resources may include NFS mounts and central processing units (CPUs). A k-stat for a particular resource contains statistical data that pertains to the particular resource. For example, a k-stat for a particular NFS mount might contain information about how much information has been written to the

particular NFS mount. For another example, a k-stat for a particular CPU might contain information about the percentage of time that the particular CPU has been idle.

[0038] In one embodiment, kernel 150 contains multiple modules, also referred to as subsystems. For example, kernel 150 might contain a remote procedure call ("RPC") module and an NFS module. In one embodiment, each system resource is associated with a module. Different system resources may be associated with different modules. In one embodiment, a module creates a separate k-stat for each system resource with which that module is associated. For each such k-stat, the module may establish an association between that k-stat and the global OSE (i.e., global zone 130) and/or one or more VOSEs (i.e., non-global zones 140) within the global OSE. Different k-stats may be associated with different zones.

[0039] For example, a process executing in non-global zone 140a may send, to kernel 150, a request to mount an NFS file system. In response to receiving the request from the process, kernel 150 may forward the request to an NFS module. In response to receiving the request from kernel 150, the NFS module may mount the NFS file system, thereby generating an NFS mount. In connection with mounting the NFS file system, the NFS module may generate a k-stat for the NFS mount, and establish an association between the k-stat and the NFS mount. The NFS module may determine in which of non-global zones 140 the process executes. In response to determining that the process executes in non-global zone 140a, the NFS module may establish an association between the NFS mount's k-stat and non-global zone 140a. The NFS module also may establish an association between the NFS mount's k-stat and global zone 130.

[0040] For another example, a particular module may generate a separate k-stat for each CPU in a plurality of CPUs. For each such CPU, the particular module may associate that CPU with any of a plurality of resource pools. Each such resource pool may comprise, for

example, CPU resources, memory resources, and/or other resources of OSE 100. A process executing in global zone 130 may send, to kernel 150, a request to bind non-global zone 140b with a particular resource pool. In response to receiving the request from the process, kernel 150 may forward the request to the particular module. In response to receiving the request from kernel 150, the particular module may establish a binding between the particular resource pool and the specified non-global zone (in this example, non-global zone 140b). In connection with establishing the binding, the particular module may establish, for each CPU in the particular resource pool, an association between that CPU's k-stat and non-global zone 140b. The particular module also may establish an association between each such CPU's k-stat and global zone 130. A process executing in a particular non-global zone will only "see" k-stats for CPUs in the resource pools to which the particular non-global zone is bound.

[0041] As a result, the k-stats of different system resources may be associated with different zones. Each k-stat may be associated with one or more zones. For example, if both non-global zone 140a and non-global zone 140b are bound to a particular resource pool, then the k-stats of the CPUs in the particular resource pool may be associated with both non-global zone 140a and non-global zone 140b.In one embodiment, either periodically or in response to the occurrence of an event relative to a system resource, kernel 150 adds to or updates statistical information pertaining to that system resource in that system resource's associated k-stat.

[0042] In one embodiment, when a process requests a list of k-stats, kernel 150 determines the zone in which the process executes. Kernel 150 selects, from among a plurality of k-stats, a set of k-stats that contains only those k-stats that are associated with the process' zone. Kernel 150 returns, to the process, a list of the k-stats in the set.

[0043] In one embodiment, kernel 150 collectively exports all k-stats as a device node from which processes can read statistical data. For example, in each zone, the device node may appear as though it were a file "/dev/kstat". When a process accesses the device node, the process may specify information that identifies a particular k-stat. In one embodiment, each k-stat is associated with a name, an instance, and a module. Thus, a process may specify a name, an instance, and a module to identify a k-stat. According to one embodiment, multiple k-stats may be associated with the same name, instance, and module, as long as each of those k-stats is associated with a different zone.

[0044] In one embodiment, when a process accesses the device node to read statistical data from a particular k-stat, the process specifies a name, instance, and module. Kernel 150 determines in which zone the process is executing. Kernel 150 selects, from among a plurality of k-stats, only those k-stats that are associated with the process-specified name, instance, and module (as is mentioned above, k-stats in different zones might be associated with the same name, instance, and module). Kernel 150 determines whether any k-stat in the selected statistical data structures is associated with the zone identifier of the zone in which the process is executing. If kernel 150 determines that such a k-stat exists in the selected statistical data structures, then kernel 150 returns, to the process, statistical information that is stored in the k-stat. If no such k-stat exists in the selected statistical data structures, then kernel 150 does not return any statistical information to the process. As a result, processes may obtain statistical information that pertains specifically to zones in which those processes execute. Additionally, processes are prevented from obtaining statistical information that pertains only to zones to which those processes do not have access.

[0045] According to one embodiment, statistical information that is stored in a k-stat may include an identity of a mount, and/or a quantity of data read from and/or written to the

mount by a process. According to one embodiment, statistical information that is stored in a k-stat may include an identity of a CPU, and/or an amount of the CPU's time used by all processes.

[0046] In one embodiment, by associating a k-stat with global zone 130 in addition to any non-global zones 140 with which that k-stat is associated, users and processes associated with global zone 130 may obtain statistical information concerning the use of system resources by users and processes associated with multiple zones in OSE 100. In one embodiment, by associating selected k-stats only with global zone 130, users and processes that are associated only with non-global zones 140 are prevented from accessing (and spared from being inundated by) statistical information stored in those selected k-stats.

Sample Operation

[0047] With the above information in mind, a sample of operation of system 100 in accordance with one embodiment of the present invention will now be described. In the following discussion, reference will be made to the system diagram of Fig. 1 and the flow diagrams of Fig. 3A and Fig. 3B.

[0048] Fig. 3A depicts an operational flow for virtualizing system statistics in an OSE that has been partitioned into a global zone and one or more non-global zones, in accordance with one embodiment of the present invention. Although Fig. 3A depicts operations being performed in a particular order, operations such as those depicted may, in various different embodiments, be performed in various different orders.

[0049] In block 302, a global zone is established under the control of a kernel. For example, global zone 130 may be established under the control of kernel 150. In block 304, one or more non-global zones, under the control of the kernel, are established within the

global zone. For example, non-global zones 140 may be established within global zone 130. Thus, the global zone and the one or more non-global zones are established within the same OSE, rather than separate OSEs having separate kernels.

[0050] In block 306, an association is established between a system resource and a k-stat. For example, kernel 150 may establish an association between an NFS mount and a k-stat that corresponds specifically to the NFS mount. Given a plurality of system resources, each such system resource may be associated with a different k-stat of a plurality of k-stats. In one embodiment, each such k-stat may be associated with a name, an instance, and a module. Although reference is made to the establishment of a single association between a single system resource and a single k-stat, many of such associations may be established, and each such association may be between a different system resource and a different k-stat. In block 308, an association is established between the k-stat and a zone. For example, kernel 150 may establish an association between an NFS mount's associated k-stat and the zone in which the NFS mount was initiated. Although reference is made to the establishment of a single association between a single k-stat and a single zone, many of such associations may be established, and each such association may be between a different k-stat and a different zone.

[0051] In block 310, a process is started in a zone. For example, a process may be started in non-global zone 140a. In block 312, in response to the starting of the process, an association is established between the process and the zone in which the process was started. For example, kernel 150 may establish an association between non-global zone 140a and a process that was started in non-global zone 140a. Given a plurality of processes, each such process may be associated with global zone 130 or any one of non-global zones 140.

[0052] In block 314, one or more key values are received from the process. For example, kernel 150 may receive, from a process executing in non-global zone 140a, key values that indicate a name, an instance, and a module. In block 316, from a plurality of k-stats, only those k-stats that are associated with the one or more process-specified key values are selected to be in a set of k-stats. For example, if the name, instance, and module is "foo, 0, foo," then kernel 150 selects, from all k-stats, a set of k-stats that are associated with "foo, 0, foo"; such k-stats may be associated with various different zones.

[0053] In block 318, it is determined whether any k-stat in the selected set of k-stats is associated with the zone identifier of the zone in which the process executes. For example, kernel 150 may determine whether any k-stat in the selected set of k-stats is associated with non-global zone 140a, in which the process executes. If a particular k-stat in the selected set is associated with the process' associated zone identifier, then control passes to block 320. Otherwise, control passes to block 322.

[0054] In block 320, statistical data stored in the particular k-stat is sent to the process. For example, kernel 150 may send, to a process executing in non-global zone 140a, statistical information that is stored in a k-stat that is associated with (a) the process-specified name, instance, module tuple and (b) non-global zone 140a. The statistical information might indicate, for example, a quantity of data that has been written to an NFS mount that is associated with the k-stat.

[0055] Alternatively, in block 322, no statistical data is sent to the process.

[0056] Fig. 3B depicts an operational flow for constructing a virtualized list of statistical data structures in an OSE that has been partitioned into a global zone and one or more non-global zones, in accordance with one embodiment of the present invention. Although Fig.

3B depicts operations being performed in a particular order, operations such as those depicted may, in various different embodiments, be performed in various different orders.

[0057] In block 352, a global zone is established under the control of a kernel. For example, global zone 130 may be established under the control of kernel 150. In block 354, one or more non-global zones, under the control of the kernel, are established within the global zone. For example, non-global zones 140 may be established within global zone 130. Thus, the global zone and the one or more non-global zones are established within the same OSE, rather than separate OSEs having separate kernels.

[0058] In block 356, an association is established between a system resource and a k-stat. For example, kernel 150 may establish an association between an NFS mount and a k-stat that corresponds specifically to the NFS mount. Although reference is made to the establishment of a single association between a single system resource and a single k-stat, many of such associations may be established, and each such association may be between a different system resource and a different k-stat. In block 358, an association is established between the k-stat and a zone. For example, kernel 150 may establish an association between an NFS mount's associated k-stat and the zone in which the NFS mount was initiated. Although reference is made to the establishment of a single association between a single k-stat and a single zone, many of such associations may be established, and each such association may be between a different k-stat and a different zone.

[0059] In block 360, a process is started in a zone. For example, a process may be started in non-global zone 140a. In block 362, in response to the starting of the process, an association is established between the process and the zone in which the process was started. For example, kernel 150 may establish an association between non-global zone 140a and a

process that was started in non-global zone 140a. Given a plurality of processes, each such process may be associated with global zone 130 or any one of non-global zones 140.

[0060] In block 364, a request for a list of k-stats is received from the process. For example, kernel 150 may receive, from a process executing in non-global zone 140a, a request for a list of k-stats. In block 366, the zone in which the process executes is determined. For example, from a zone identifier that was associated with a process in conjunction with the starting of the process, kernel 150 may determine that the process executes in non-global zone 140a.

[0061] In block 368, from a plurality of k-stats, only those k-stats that are associated with the zone identifier of the zone in which the process executes are selected to be in a set of k-stats. For example, if the process executes in non-global zone 140a, then only those k-stats that also are associated with the zone identifier of non-global zone 140a are selected to be in the set. In block 370, a list of the k-stats in the selected set is sent to the process. For example, kernel 150 may send, to a process executing in non-global zone 140a, a list of k-stats that includes only k-stats that are associated with non-global zone 140a, and all of such k-stats. The list may identify the k-stats without revealing the contents of those k-stats.

Hardware Overview

[0062] In one embodiment, the various components of computing environment 100 shown in Fig. 1 can be implemented as sets of instructions executable by one or more processors. These components may be implemented as part of an operating system, including but not limited to the Solaris™ operating system produced by Sun Microsystems, Inc. Figure 4 is a block diagram that illustrates a computer system 400 upon which an embodiment of the invention may be implemented. Computer system 400 includes a bus 402 for facilitating information exchange, and one or more processors 404 coupled with bus 402

for processing information. Computer system 400 also includes a main memory 406, such as a random access memory (RAM) or other dynamic storage device, coupled to bus 402 for storing information and instructions to be executed by processor 404. Main memory 406 also may be used for storing temporary variables or other intermediate information during execution of instructions by processor 404. Computer system 400 may further include a read only memory (ROM) 408 or other static storage device coupled to bus 402 for storing static information and instructions for processor 404. A storage device 410, such as a magnetic disk or optical disk, is provided and coupled to bus 402 for storing information and instructions.

[0063] Computer system 400 may be coupled via bus 402 to a display 412, such as a cathode ray tube (CRT), for displaying information to a computer user. An input device 414, including alphanumeric and other keys, is coupled to bus 402 for communicating information and command selections to processor 404. Another type of user input device is cursor control 416, such as a mouse, a trackball, or cursor direction keys for communicating direction information and command selections to processor 404 and for controlling cursor movement on display 412. This input device typically has two degrees of freedom in two axes, a first axis (e.g., x) and a second axis (e.g., y), that allows the device to specify positions in a plane.

[0064] In computer system 400, bus 402 may be any mechanism and/or medium that enables information, signals, data, etc., to be exchanged between the various components. For example, bus 402 may be a set of conductors that carries electrical signals. Bus 402 may also be a wireless medium (e.g. air) that carries wireless signals between one or more of the components. Bus 402 may also be a medium (e.g. air) that enables signals to be capacitively exchanged between one or more of the components. Bus 402 may further be a network

connection that connects one or more of the components. Overall, any mechanism and/or medium that enables information, signals, data, etc., to be exchanged between the various components may be used as bus 402.

[0065] Bus 402 may also be a combination of these mechanisms/media. For example, processor 404 may communicate with storage device 410 wirelessly. In such a case, the bus 402, from the standpoint of processor 404 and storage device 410, would be a wireless medium, such as air. Further, processor 404 may communicate with ROM 408 capacitively. In this instance, the bus 402 would be the medium (such as air) that enables this capacitive communication to take place. Further, processor 404 may communicate with main memory 406 via a network connection. In this case, the bus 402 would be the network connection. Further, processor 404 may communicate with display 412 via a set of conductors. In this instance, the bus 402 would be the set of conductors. Thus, depending upon how the various components communicate with each other, bus 402 may take on different forms. Bus 402, as shown in Fig. 4, functionally represents all of the mechanisms and/or media that enable information, signals, data, etc., to be exchanged between the various components.

[0066] The invention is related to the use of computer system 400 for implementing the techniques described herein. According to one embodiment of the invention, those techniques are performed by computer system 400 in response to processor 404 executing one or more sequences of one or more instructions contained in main memory 406. Such instructions may be read into main memory 406 from another machine-readable medium, such as storage device 410. Execution of the sequences of instructions contained in main memory 406 causes processor 404 to perform the process steps described herein. In alternative embodiments, hard-wired circuitry may be used in place of or in combination with

software instructions to implement the invention. Thus, embodiments of the invention are not limited to any specific combination of hardware circuitry and software.

[0067] The term "machine-readable medium" as used herein refers to any medium that participates in providing data that causes a machine to operation in a specific fashion. In an embodiment implemented using computer system 400, various machine-readable media are involved, for example, in providing instructions to processor 404 for execution. Such a medium may take many forms, including but not limited to, non-volatile media, volatile media, and transmission media. Non-volatile media includes, for example, optical or magnetic disks, such as storage device 410. Volatile media includes dynamic memory, such as main memory 406. Transmission media includes coaxial cables, copper wire and fiber optics, including the wires that comprise bus 402. Transmission media can also take the form of acoustic or light waves, such as those generated during radio-wave and infra-red data communications.

[0068] Common forms of machine-readable media include, for example, a floppy disk, a flexible disk, hard disk, magnetic tape, or any other magnetic medium, a CD-ROM, any other optical medium, punchcards, papertape, any other physical medium with patterns of holes, a RAM, a PROM, and EPROM, a FLASH-EPROM, any other memory chip or cartridge, a carrier wave as described hereinafter, or any other medium from which a computer can read.

[0069] Various forms of machine-readable media may be involved in carrying one or more sequences of one or more instructions to processor 404 for execution. For example, the instructions may initially be carried on a magnetic disk of a remote computer. The remote computer can load the instructions into its dynamic memory and send the instructions over a telephone line using a modem. A modem local to computer system 400 can receive the data on the telephone line and use an infra-red transmitter to convert the data to an infra-red

signal. An infra-red detector can receive the data carried in the infra-red signal and appropriate circuitry can place the data on bus 402. Bus 402 carries the data to main memory 406, from which processor 404 retrieves and executes the instructions. The instructions received by main memory 406 may optionally be stored on storage device 410 either before or after execution by processor 404.

[0070] Computer system 400 also includes a communication interface 418 coupled to bus 402. Communication interface 418 provides a two-way data communication coupling to a network link 420 that is connected to a local network 422. For example, communication interface 418 may be an integrated services digital network (ISDN) card or a modem to provide a data communication connection to a corresponding type of telephone line. As another example, communication interface 418 may be a local area network (LAN) card to provide a data communication connection to a compatible LAN. Wireless links may also be implemented. In any such implementation, communication interface 418 sends and receives electrical, electromagnetic or optical signals that carry digital data streams representing various types of information.

[0071] Network link 420 typically provides data communication through one or more networks to other data devices. For example, network link 420 may provide a connection through local network 422 to a host computer 424 or to data equipment operated by an Internet Service Provider (ISP) 426. ISP 426 in turn provides data communication services through the world wide packet data communication network now commonly referred to as the "Internet" 428. Local network 422 and Internet 428 both use electrical, electromagnetic or optical signals that carry digital data streams. The signals through the various networks and the signals on network link 420 and through communication interface 418, which carry

the digital data to and from computer system 400, are exemplary forms of carrier waves transporting the information.

[0072] Computer system 400 can send messages and receive data, including program code, through the network(s), network link 420 and communication interface 418. In the Internet example, a server 430 might transmit a requested code for an application program through Internet 428, ISP 426, local network 422 and communication interface 418. The received code may be executed by processor 404 as it is received, and/or stored in storage device 410, or other non-volatile storage for later execution. In this manner, computer system 400 may obtain application code in the form of a carrier wave.

[0073] In the foregoing specification, embodiments of the invention have been described with reference to numerous specific details that may vary from implementation to implementation. Thus, the sole and exclusive indicator of what is the invention, and is intended by the applicants to be the invention, is the set of claims that issue from this application, in the specific form in which such claims issue, including any subsequent correction. Any definitions expressly set forth herein for terms contained in such claims shall govern the meaning of such terms as used in the claims. Hence, no limitation, element, property, feature, advantage or attribute that is not expressly recited in a claim should limit the scope of such claim in any way. The specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense.
